UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/693,633	10/23/2003	Joseph S. Beda	3481	8889

7590 01/24/2005

Law Offices of Albert S. Michalik, PLLC
Suite 193
704 -228th Avenue NE
Sammamish, WA 98074

EXAMINER	
WOODS, ERIC V	
ART UNIT	**PAPER NUMBER**
2672	

DATE MAILED: 01/24/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/693,633	BEDA ET AL.	
	Examiner	Art Unit	
	Eric V Woods	2672	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *23 October 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-63* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-63* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *23 October 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☒ Interview Summary (PTO-413) Paper No(s)/Mail Date. *20050119* .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Specification

1. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

Claim Objections

2. The numbering of claims is not in accordance with 37 CFR 1.126 which requires the original numbering of the claims to be preserved throughout the prosecution. When claims are canceled, the remaining claims must not be renumbered. When new claims are presented, they must be numbered consecutively beginning with the number next following the highest numbered claims previously presented (whether entered or not).

3. Claim 65 objected to because of the following informalities: it is numbered as claim 65 but should be claim 63. Appropriate correction is required. For purposes of further examination, claim 65 will be termed claim 63.

4. Claim 37 is objected to because it is dependent on claim 26, and it is placed after claim 36. It is unknown whether or not this was intended, but the claim numbering is wrong, in that claims that are dependent on a previous independent claim should not be listed after a new independent claim, which is confusing.

5. Claims 56-65 are objected to because there is no claim 55. The claim numbering should be corrected to remedy this problem. Appropriate correction is required.

6. Claim 57 is objected to because in the third line of the claim there is a letter 'd' by itself with no meaning associated with it. Examiner will ignore it for purposes of examination. Appropriate correction is required.

Claim Rejections - 35 USC § 101

7. 35 U.S.C. 101 reads as follows:

> **Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.**

8. Claims 1-62 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. That is, the method claims are clearly software as supported by applicant's specification and they clearly read on a software implementation. Such an implementation or a computer program is never claimed. As such, the claims in question (independent claims 1 and 36, and all dependent claims thereof) recite functional descriptive material, that is, software per se, and as such are prima facie nonstatutory. Further, claims 1 and 36 are not technologically embodied, as a "computing environment" in claim 1 could be a room with a computing device in it. See MPEP 2106 and *In re Prater*.

To expedite a complete examination of the instant application, the claims rejected above under 35 U.S.C. 101 (nonstatutory) are further rejected as set forth below in anticipation of applicant amending the claims to place them within the four statutory categories of invention.

Claim Rejections - 35 USC § 112

9. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

10. Claims 6-65 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

11. Specifically, claims 6-65 are rejected because the term "a visual" is used where it is unclear what the terminology is referring to and the specification does not reasonably apprise one of ordinary skill in the art what the definition would be. **For examination purposes, examiner will treat "a visual" as meaning "a visual object or element" in the context of a visual element in SVG specification, which applicant clearly states is being used in his invention.

12. Claim 15 is rejected because the term 'path' is used, and it is unclear what the meaning of 'path' – whether applicant is referring to the path of a visual element during an animation or a path as defined in the SVG specification.

13. Claim 20 is rejected because the term 'interface' is used, and it is unclear what the intended meaning of 'interface' is in the context of the specification, because applicant refers to the SVG specification, wherein an 'interface' is a programming construct – that is, an interface implemented for elements, wherein it is unclear whether that is intended meaning or a generalized interface (hardware or software) is intended.

14. Claim 4, 26, and 27 are rejected because the term 'context' is used, and it is unclear what the meaning of 'context' is in this claim. In the SVG specification, the term 'context' is not specifically set forth and applicant does not provide a clear basis of the definition. Context can relate to the device-specific drawing information and

capabilities, as set forth by various references; it can be a term relating specifically to the properties and metadata associated with various visual elements; it can be many things.

15. Claims 37 and 40 are rejected because they should be dependent on claim 36 as per examiner's conversation with attorney of record Albert Michalik.

16. Claim 39 is rejected because the term 'collection object' is used, and it is unknown how such an object would differ from the 'container object' recited in the parent claim 36 to this particular claim and the specification does not offer a separate definition for such an object. The definition adopted by the examiner is listed in the rejected to the claim.

Claim Rejections - 35 USC § 102

17. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> **A person shall be entitled to a patent unless –**
>
> **(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.**

18. Claims 1 and 36 are rejected under 35 U.S.C. 102(b) as being anticipated by a mental process in a human being augmented with a pencil and paper (See MPEP 2106 for *In re Prater*). As to claim 1, a "computing environment" in claim 1 could be a room with a computing device in it. The recited activities, such as receiving a function call via an interface, could be received by a human being as a piece of paper (or as IPvA – that is, IP written on paper over avian carrier or some other means), and the resultant

alteration of data in a scene graph could be performed on a piece of paper. The same logic holds for claim 36.

Dependent claims 2-35 and 37-62 are so rejected for not correcting the deficiencies of their parent claims. Amending the above claims to be technologically embodied and not recite software per se, as required in the above rejection under 35 U.S.C. 101, will cause the removal of this rejection under 35 U.S.C. 102.

Claim Rejections - 35 USC § 103

19. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> **(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.**

20. Claims 1-28 and 30-65 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kim et al (US PGPub 2003/0120823 A1)('Kim') in view of Steele et al (US PGPub 2004/0110490 A1)('Steele') further in view of SVG (Scalable Vector Graphics) specification version 1.1, given hereafter as 'SVG' (revision 1.1, 14 January 2003)). **As established before, applicant's invention appears to be directed to an extension of the SVG specification, at least partially, thus making it a valid citation (see rejection to claim 2 for citations of applicant's specification where this is cited, specifically).

**Examiner is using the following definition of "scene graph", which consists of objects, called nodes, arranged in a tree structure that does not contain cycles. (Taken from Sun's Java documentation (see attached – termed 'Sun'))(E.g. directed acyclic

graph (DAG) and NIST's website for the definition of a tree graph as a generalized directed acyclic graph).

21. As to claim 1,

In a computing environment a method comprising,

-Receiving a function call via an interface, the function call comprising markup language data; and (Kim Fig. 2 clearly shows that the system receives 3D data in X3D format, which is known to be in a markup language [0004-0008]. X3D is known to one of ordinary skill in the art to be the next generation of the VRML (virtual reality markup language) and to accept and be an extension of XML (extensible markup language). Next, the X3D browser – element 140 in Figs. 1 and 2, makes function call based on information that it obtains when it sends out requests for data. Further, see Fig. 3, where clearly the system is shown to receive user events and data from the user interface. Clearly, these represent function calls via an interface.) (Steele Fig. 2, element 210 converter – see Fig. 3 for enlarged version, with SVG conversion shown in Fig. 6, SVG is *prima facie* a markup language [see 0041]. Clearly, the converter receives function calls through the interface, e.g. the application 205 sends function calls to the converter for graphics data, which are then routed through the converter 210 – again, it is a fundamental of the computer art that applications sending data through other programs, particularly in this kind of a context.)

-Interpreting the markup language data to cause data in a scene graph to be modified. (Kim [0007-0008] clearly teaches the use of a scene graph and that X3D requires the construction of such scene graphs from primitives. Kim further teaches that the user

can move through a scene [0020, 0026], which clearly establishes that a user is navigating and the scene is constantly being re-rendered, which *prima facie* requires data in the scene graph to be modified. Kim can also use MPEG-4, which clearly involves animation and modification of data in a scene graph.)(Steele uses SVG, and teaches the decomposition of graphic data into tree structures in Fig. 7. Further, Fig. 8 clearly illustrates what happens when the SVG animation language is transformed into two sets of output language data. Clearly, as Fig. 9 illustrates, animation is done with SVG on a routine basis and the translation is shown in Fig. 6. The tree of Fig. 7 is clearly a form of scene graph in the broad definition set forth above. Clearly, such a tree is a "scene graph". Thusly and *prima facie* obviously, the animation shown in Fig. 9 would cause data in a scene graph to be modified as the object was translated and the data structure containing locations and other information would changed.)

Thusly, reference Kim teaches all the limitations of the above claim, but does not expressly teach the modification of data in the scene graph, although, as set forth above, such functionality is inherent in the reference. Reference Steele clearly teaches how multimedia is converted and formatted for use for multiple devices. Reference Kim teaches the use of client 100 with X3D browser 110 over communication network 150 [0015, 0018, 0020]. Clearly, a portable device could fulfill this device (e.g. advanced PDA, cellular phone, or laptop.) Reference Steele teaches portable devices capable of performing advanced graphics functionality such as recited above in [0007-0008] and [0027], which clearly illustrates the invention functioning on a cellular phone in Fig. 27. Further, Steele teaches that his invention can be practiced on portable devices

(including computers [0038]) – see Fig. 1 and the devices 105, which would clearly be the clients 100 of Kim and the communications networks would be comparable. Clearly, the references are directed to the same problem-solving area and further as set forth above are analogous art. As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize.

**Note: Steele clearly teaches the modification of data, including animation, et cetera in [0061].

22. As to claim 2,

The method of claim 1 wherein causing data in the scene graph to be modified comprises causing initialization of a new instance of a visual class.

Reference Kim does not expressly teach this limitation, but implicitly does because of the use the X3D specification. This specification – and the structure of markup languages in general – means that when non-visible objects become visible, e.g. data is pulled from the X3D stream that does so through the parser [0020, 0025-0026], the data in the scene graph changes as well, as with limited memory the entire 3D world cannot be stored on the client. The same logic applies to the Steele reference, which teaches that SVG data is decomposed into scene graphs, a.k.a. trees, (see Figure 7), and again – whenever new visual elements enter the scene, new subgroups are instantiated, which *prima facie* (see SVG specification, section 9) are

elements that compose visual objects, which therefore are new instances of a visual class as recited above, since a class as recited by applicant is comparable to the basic 'shapes' in SVG (applicant's specification clearly uses it – 23:1-20 where applicant's invention adopts all classes and shapes from SVG) and thusly meets the recited limitation. As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize. (See claim 1 for additional motivation / combination justification, which is herein incorporated by reference).

23. As to claim 3,

The method of claim 2 wherein causing data in the scene graph to be modified comprises invoking code to associate a transform with a visual object in the scene graph.

Reference Kim does not expressly teach this limitation, but clearly teaches that users navigate through a virtual environment, which *prima facie* requires that transforms take place to visual objects. Reference Steele teaches this limitation, as he discloses rotations in [0053] and further states that rotations and other transformations can be applied to an entire tree of objects, e.g. Fig. 7, and further [0088] that any visual element or object can modified. Such modifications *prima facie* must associate code with a suitable / desired transform (e.g. scaling, rotation, et cetera [0053]), as that is the

only way either a hierarchy of nodes (e.g. Fig. 7) or single nodes could be scaled. As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG specification, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize. (Please see claim 1 for additional motivation / combination justification, which is herein incorporated by reference). (Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task.)

24. As to claim 4,

The method of claim 1 wherein causing data in a scene graph data structure to be modified comprises invoking code to place a drawing visual into the scene graph.

Reference Kim does not expressly teach this limitation, but clearly teaches that users navigate through a virtual environment, which *prima facie* requires that transforms take place to visual objects and new visual objects be inserted (see rejections to claim 2 and 3). Reference Steele teaches this limitation, as for example he teaches the insertion of unique identifiers into media streams [0106], and further [0088] that any visual element or object can modified. Such modifications and insertions *prima facie* must associate code with a suitable / desired insertion as that is the only way either a hierarchy of nodes (e.g. Fig. 7) or single nodes could be logically inserted.

For the second case, if the definition of context is the data associated with a specific element – e.g. the details of the element, its location, color, et cetera, these

attributes are inherent in SVG elements as set forth in the rejections above, e.g. sections 11.1, 9.1-9.7, et cetera. Further, Steele teaches the same in Figure 7, where each element has certain properties that would be a drawing context, in the sense that each visual element has those properties associated with it [Steele 0052-0056 and 0059-0061].

As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize. (See claim 1 for additional motivation / combination justification, which is herein incorporated by reference).). (Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task.)

25. As to claim 5,

The method of claim 4 further comprising, causing drawing context to be returned, the drawing context providing a mechanism for rendering into the drawing visual.

Reference Kim does not expressly teach this limitation, but clearly teaches that users navigate through a virtual environment, which *prima facie* requires that device specific information regarding display information, e.g. drawing context, be available to the rendering engine. Reference Steele teaches this limitation, as for example he teaches the retrieval of device context in [0101]. Clearly, the device receives information based on its device context, which clearly is associated with the drawing

context, as the two are one and the same in this case. Steele teaches rendering in [0007 and 0011-0012]. The drawing context per se is incorporated into the data structures of Steele (see Figure 7). Reference Kim clearly teaches rendering in Fig. 2, element 143, "rendering means". As such, it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize. (See claim 1 for additional motivation / combination justification, which is herein incorporated by reference). It further would have been obvious to modify the system of Kim to utilize a device specific context so as to optimize data streamed to that device for purposes of minimizing memory consumption (a large problem pointed out by Steele [0007]).). (Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task.)

26. As to claim 6,

The method of claim 2 wherein causing data in the scene graph to be modified comprises invoking code to associate brush data with a visual object in the scene graph.

Reference Kim does not expressly teach this limitation. Reference Steele does teach this limitation by the use of SVG graphics. Turning to the SVG (, section 11 titled 'Painting: Filling, Stroking, and Marker Symbols', specifically section 11.1, 'With SVG, you can paint (e.g. stroke or fill) with: ..." and then proceeds to list several. The term

'brush data' is clearly analogous to the 'paint' operation in SVG with comparable data. Given that SVG allows (from section 11.1) a single color, a solid color (with or without opacity), a gradient, a pattern (vector or image), and custom patterns, clearly each visible element clearly has such data associated with it (see section 11.2 in its entirety, 11.7 for specific properties, section 11.8 for how painting properties can be inherited, which *prima facie* justifies the position that element have intrinsic painting properties, i.e. brush data as set forth above. Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task, and SVG data element *prima facie* and inherently possess paint data as set forth by the SVG specification above. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize, and the SVG standard inherently handles these paint limitations.

27. As to claim 7,

The method of claim 6 wherein the brush data comprises receiving data corresponding to a solid color.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Section 11.1 of the SVG specification sets forth that a

user can paint with a solid color with opacity, thus meeting this limitation. Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task, and SVG data element *prima facie* and inherently possess paint data as set forth by the SVG specification above. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 6 for additional motivation).

28. As to claim 8,

The method of claim 6 wherein receiving brush data comprises receiving data corresponding to a linear gradient brush and a stop collection comprising at least one stop.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Section 11.1 of the SVG specification sets forth that a user can paint with a gradient that can be linear. Further, sections 11.7.1 and 11.7.2 of the specification sets forth that gradient stops are included in the SVG 'color-interpolation' property. Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task, and SVG data element *prima facie* and inherently possess paint data as set forth by the SVG specification above. As such, reference Steele intrinsically teaches this limitation

and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 6 for additional motivation).

29. As to claim 9,

The method of claim 6 wherein receiving brush data comprises receiving data corresponding a radial gradient brush.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Section 11.1 of the SVG specification sets forth that a user can paint with a gradient that can be radial and also see sections 11.7.1 and 11.7.2 for more detail, thus meeting this limitation. Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task, and SVG data element *prima facie* and inherently possess paint data as set forth by the SVG specification above. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 6 for additional motivation).

30. As to claim 10,

The method of claim 6 wherein receiving brush data comprises receiving data corresponding to an image.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Section 11.1 of the SVG specification sets forth that a user can paint with an image with further details provided in section 11.7.5 under the 'image-rendering' property. Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task, and SVG data element *prima facie* and inherently possess paint data as set forth by the SVG specification above. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 6 for additional motivation).

31. As to claim 11,

The method of claim 10 further comprising, receiving markup corresponding to an image effect to apply to the image.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Section 14.4 of the SVG specification sets forth that a user can use any image as an opacity mask, thus meeting this limitation, given that alpha blending is *prima facie* an image effect. Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task, and SVG data element *prima facie* and inherently possess paint data

as set forth by the SVG specification above, and SVG is inherently a markup language, so the rendering portion of Steele would receive such information (the rendering functionality is inherent in SVG – see section 11.7, 14.4, et cetera). As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 6 for additional motivation).

32. As to claim 12,

The method of claim 1 further comprising, receiving markup corresponding to pen data that defines an outline of a shape.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 1 and reference SVG clearly supports this position. The term 'pen data' used by applicant above is comparable or analogous to any set of data defining the outline of a shape, including SVG 'path' data. Section 11.3 of the SVG specification sets forth that a user can fill a path that would correspond to the outline of shape with multiple illustrations provided for this under the 'nonzero' and 'even odd' subheadings – see details on paths -- with further details provided in section 11.3 and 11.4 (the individual strokes that create these effects. SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the

invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claims 1 and 2 for additional motivation).

33. As to claim 13,

The method of claim 1 wherein the markup corresponds to a shape class comprising at least one of the set containing rectangle, polyline, polygon, path, line and ellipse shapes.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 1 and reference SVG clearly supports this position. The SVG specification sets forth classes of shapes in section 9.1, where all six of the recited shapes (rectangle, polygon, path, line, polyline, and ellipse) are set forth. Further, the SVG view in Steele decomposes SVG instructions into scene graphs containing basic SVG shapes as above [Steele 0052], where 'Visual Elements' include Shape classes. SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claims 1 and 2 for additional motivation).

34. As to claim 14,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a geometry-related function to represent a rectangle in the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. The SVG specification clearly shows in section 9.1 that rectangles are a basic shape, and further that in 9.2 under Example rect02 that such rectangles can have rounded corners, and code is provided that implements such. Also, the 'Rect' class inherently has geometry-related functions as set forth in the beginning to section 9.2. SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup. As such, reference Steele shows a rectangle 715 in the scene graph in Figure 7 that intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 2 for additional motivation). Also, said element can be animated under SVG section 19.2. Steele clearly teaches data modification in [0061] as set forth above.

35. As to claim 15,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a geometry-related function to represent a path in the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Further, Steele Fig. 6 shows an animation where path information is extracted into element 620, and listed in Fig. 7 [see Steele 0050 and 0079]. Further, the SVG specification sets forth path data in section 9.1 as existing and how a 'path' can define a shape or similar. Both meanings are covered herein. Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 2 for additional motivation). Also, said element can be animated under SVG section 19.2.

36. As to claim 16,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a geometry-related function to represent a line in the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Further, Steele Fig. 6 shows an animation where path

information is extracted into element 620, and listed in Fig. 7 [see Steele 0050 and 0079]. A line element can be animated under SVG section 19.2, which is obviously geometric. Line elements are set forth in SVG section 9.5, and their geometric functions. Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality, and the scene graph shown in Figure 7 could clearly include lines since they are Visual Elements [Steele 0060-0061, which supports animation, et cetera]. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 2 for additional motivation).

37. As to claim 17,

The method of claim 1 wherein the markup is related to hit-testing a visual in the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Clearly, Kim teaches three-dimensional navigation in Figure 4 – that is, navigation using a UI through a two-dimensional view-port of a three-dimensional environment, which is what any display normally shows. Therefore, given that a portable computer could clearly be used, the user would clearly be interacting

with the display. As such, hit testing would be required for user interactivity, as could the system of Steele under the same rationale. The SVG specification sets forth hit testing in section 16.6 (the two paragraphs right at the end of the section) where hit testing (namely, hit detection) is taught, specifically testing text for character or cell hits and testing visual elements for hits in their entirety, and such information is clearly communicated in markup language – see section 16.2 for event types and elements transmitted in markup, for example. Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 2 for additional motivation).

38. As to claim 18,

The method of claim 1 wherein causing data in a scene graph data structure to be modified comprises invoking a function related to transforming coordinates of a visual in the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Further, Steele Fig. 6 shows an animation where path

information is extracted into element 620, and listed in Fig. 7 [see Steele 0050 and 0079]. Clearly, SVG teaches the animation of visual elements, see section 19.2, which *prima facie* involves transforming coordinates of a visual in the scene graph data, and according to Steele [0052-0053] and a tree of elements can also be transformed [Steele 0052]. Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality, and the scene graph shown in Figure 7 could clearly include lines since they are Visual Elements [Steele 0060-0061, which supports animation, et cetera]. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

39. As to claim 19,

The method of claim 1 wherein the markup is related to calculating a bounding box of a visual in the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Bounding box calculations are taught in section 7.1 and detailed in section 7.11 where they are calculated. Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality, and the scene graph shown in Figure 7 could clearly include lines since they are Visual Elements [Steele 0060-0061, which supports animation, et cetera and would logically include bounding boxes]. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

40. As to claim 20,

The method of claim 1 wherein causing data in the scene graph be modified comprises invoking a function via a common interface to a visual object in the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Clearly, the SVG specification teaches interfaces in section 4.3 – there are common DOM interface as set forth there. If the intended meaning of applicant was that such interfaces were based in hardware or software, fundamentally in reference Kim the user interacts with the X3D browser that would provide a common interface, in that all events generated by such browser would go to an interface – that is, they would move through Kim's communication module 130 in Fig. 1 to the applet 110, which would be an interface per se, and reference Steele clearly

sets forth that his invention has various possible interfaces, depending on the embodiment (e.g. PDA, cell phone, et cetera [0004] and [0007-0008]).

Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality, and the scene graph shown in Figure 7 could clearly include lines since they are Visual Elements [Steele 0060-0061, which supports animation, et cetera]. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

41. As to claim 21,

The method of claim 1 further comprising invoking a visual manager to render a tree of at least one visual object to a rendering target.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Further, Steele Fig. 6 shows an animation where path information is extracted into element 620, and listed in Fig. 7 [see Steele 0050 and 0079] as trees. SVG teaches that all implementations must implement a rendering model as set forth in 3.1 and so forth, and scene graphs are known to directed acyclic, i.e. trees. Clearly, this model is implemented through the DOM interfaces set forth in

section 4, and each element has its own element information that controls rendering aspects. Steele clearly teaches data modification in [0061] as set forth above. It is *prima facie* obvious that a 'visual manager' of some form must exist in order to handle formatting issues and precedence in rendering, and Steele teaches such a manager in [0075-0076. Clearly the rendering information each visual element [Steele 0056-0061] is sufficient such that it is its own 'rendering target' as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality, and the scene graph shown in Figure 7 could clearly include lines since they are Visual Elements [Steele 0060-0061, which supports animation, et cetera]. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation). It would have been obvious to one of ordinary skill in the art to add a manager if necessary to determine the precedence of item rendering with respect to the tree.

42. As to claim 22,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a function to place a container object in the scene graph data structure, the contained object configured to contain at least one visual object.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Further, Steele Fig. 7 shows a tree derived from an animation is shown – Figure 6 [see Steele 0050 and 0079]. SVG clearly teaches the use of container objects, as in section 1.6 it clearly teaches the use of 'container elements', which are defined as 'An element that can have graphic elements and other container elements as child elements'. Steele clearly teaches data modification in [0061] as set forth above. Clearly, the container object could be the head object of the tree structure shown in Steele Fig. 7.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality, and clearly the addition of such an element to the viewable area in Steele would create it. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

43. As to claim 23,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a function to place image data into the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG

clearly supports this position. Clearly, visual elements can be covered by or tiled with images as established in SVG section 11.1, where SVG teaches: "...can paint (i.e. fill or stroke) with: ...a pattern (vector or image, possibly tiled) ..." which clearly establishes this, with more detail in section 11.7.5 and 11.12.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

44. As to claim 24,

The method of claim 23 wherein causing data in the scene graph to be modified comprises invoking a function to place an image effect object into the scene graph data structure that is associated with the image data.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Section 14.4 of the SVG specification sets forth that a user can use any image as an opacity mask for any visual element, thus meeting this limitation, given that alpha blending is *prima facie* an image effect. Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

45. As to claim 25,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a function to place data corresponding to text into the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Section 10.1 of the SVG specification sets forth the use of a 'text' element, and Steele teaches the inclusion of text element 725 in the data tree shown in Fig. 7. Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics

system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

46. As to claim 26,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a function to provide a drawing context in response to the function call.

Reference Kim does not expressly teach this limitation, but clearly teaches that users navigate through a virtual environment, which *prima facie* requires that device specific information regarding display information, e.g. drawing context, be available to the rendering engine. Reference Steele teaches this limitation, as for example he teaches the retrieval of device context in [0101]. Clearly, the device receives information based on its device context, which clearly is associated with the drawing context, as the two are one and the same in this case. For the second case, if the definition of context is the data associated with a specific element – e.g. the details of the element, its location, color, et cetera, these attributes are inherent in SVG elements as set forth in the rejections above, e.g. sections 11.1, 9.1-9.7, et cetera. Further, Steele teaches the same in Figure 7, where each element has certain properties that would be a drawing context, in the sense that each visual element has those properties associated with it [Steele 0052-0056 and 0059-0061].

Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such

functionality. SVG is also a subset of XML, and SVG teaches metadata use in section 21.1. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

It further would have been obvious to modify the system of Kim to utilize a device specific context so as to optimize data streamed to that device for purposes of minimizing memory consumption (a large problem pointed out by Steele [0007]), and the SVG DOM interfaces in section 4.1-4.4 (SVG specification) clearly provide methods for retrieving drawing information, which would be that context.

47. As to claim 27,

The method of claim 26 wherein the function call corresponds to a retained visual, and further comprising, calling back to have the drawing context of the retained visual returned to the scene graph data structure.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 6 and reference SVG clearly supports this position. Section 11.1 of the SVG specification clearly sets forth that all elements (as well as 3.1 and 4.2) have properties associated with them. The system of Steele clearly caches visuals during processing – see [0083], and it would be obvious that such data would be pulled from the cache to find out the state and

properties of a visual element. Steele clearly teaches data modification in [0061] as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation). Further, it would be obvious to one of ordinary skill to so modify the Kim reference to cache the visuals so that they would be retained and that data would be pulled from the cache as set forth above, and as the Steele reference sets forth to have it pulled from there during data processing, including that of data trees like unto the one in Figure 7, as in [0100 Steele].

48. As to claim 28,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a function to place a three-dimensional visual into the scene graph data structure.

The Kim reference clearly teaches this limitation. The Kim reference clearly teaches scene graphs as established in the rejection to claim 1 [0007-0009]. Clearly, Kim teaches the use of three-dimensional data under the X3D standard specification, which is a form of XML [0007-0009]. Clearly, Kim teaches [0032-0034] that scene data is processed and all objects have specific sets of data associated with them, for

example [0042-0044], which clearly establishes that all objects have three-dimensional attributes and properties. This *prima facie* establishes that three-dimensional visuals are placed into the scene graph data structure. Clearly, the system implements the X3D specification in software, and, as such, it is software, which *prima facie* uses function calls. As such, only the primary reference is utilized and no separate motivation or combination is required; if necessary, that of the rejection to the parent claim is herein incorporated via reference.

49. Claim 29 is rejected under 35 U.S.C. 103(a) as unpatentable over Kim in view of Steele and SVG as applied to claim 28 above, and further in view of X3D (X3D specification).

As to claim 29,

The method of claim 28 wherein causing data in the scene graph to be modified comprises mapping a two-dimensional surface onto the three dimensional visual.

The Kim reference clearly teaches this limitation, and X3D standard is only cited to clarify certain points. The Kim reference clearly teaches scene graphs as established in the rejection to claim 1 [0007-0009]. Clearly, Kim teaches the use of three-dimensional data under the X3D standard specification, which is a form of XML [0007-0009]. Clearly, Kim teaches [0032-0034] that scene data is processed and all objects have specific sets of data associated with them, for example [0042-0044], which clearly establishes that all objects have three-dimensional attributes and properties. This *prima facie* establishes that three-dimensional visuals are placed into the scene graph data structure. Clearly, the system implements the X3D specification in software, and, as

such, it is software, which *prima facie* uses function calls. Secondly, the X3D standard clearly allows for the incorporation of 2D images onto three-dimensional elements, as stated in X3D 18.2.1 and 18.4.1, particularly 18.4.1, which reads clearly that "browsers may support other image formats ... which may be rendered into a 2D image" and clearly those images can be applied to three-dimensional objects such as those described in 18.3.1 and as defined in 18.2.1-18.2.3.

As such, only the primary reference is utilized and no separate motivation or combination is required; if necessary, that of the rejection to the parent claim is herein incorporated via reference. No separate motivation is required for the use of the X3D standard, as reference Kim explicitly states over 200 times that that standard is the one being used and it is central to his invention. Motivation and combination, if required, are adopted from the rejection to the parent claim and hereby incorporated via reference.

50. As to claim 30,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a function to place animation data into the scene graph data structure.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Steele in Figs. 6 and 9 shows animated elements [0041] and in Fig. 7 shows that each subgroup is shifted a certain amount with x and y coordinates given. Steele [0050, 0052] for example provides that such animation takes place, and the SVG standard in 19.1 – 19.5 clearly sets forth how each element can have animation associated with it, which clearly is placed into the scene graph of Fig. 7. Therefore,

clearly animation data is put into the tree of Fig. 7 Steele, which is clearly a scene graph by every known definition of the term, and a sample SVG XML program is provided in the second page of Fig. 9.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

51. As to claim 31,

The method of claim 30 further comprising communicating timeline information corresponding to the animation data to a composition engine.

References Kim does not expressly teach this limitation – reference Kim does teach rendering (143) and scene processing (144) means in Fig. 2, which clearly can perform compositing (since the system does 3D rendering), while reference Steele does teach the animation limitation. Steele clearly establishes in [0051-0054] and Figs. 6 and 9 that animation takes place through the SVG standard. Section 19.2 of SVG sets forth how this takes place, and at the bottom three paragraphs of section 19.2.2 it clearly states that animati0on has a document start and document end, and further in the second to last paragraph that the SVG system indicates the timeline position of document fragments being animated. Further, according to SVG 19.2.2 the animation

is by document fragment and object path, which clearly are passed to the system is specified in, for example, the second page of Fig. 9 in the SVG XML program. Clearly, the system of Steele performs compositing and rendering [0007, 0011-0012], as does Kim. Finally, reference SVG teaches that it supports compositing (section 14.2.1), and the X3D specification supports compositing (6.2.3 for example). The composition engine would be, for example, elements 143, 144, and 147 of Fig. 2 of Kim.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

52. As to claim 32,

The method of claim 31 wherein the composition engine interpolates graphics data based on the timeline to animate an output corresponding to an object in the scene graph data structure.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Steele in Figs. 6 and 9 shows animated elements [0041] and in Fig. 8, clearly during an animation the actions are shown, where the system in steps 835, 840, and 845 performs interpolation for nodes shown in the tree in Fig. 7. Clearly interpolation takes place during animation [0072, 0077-0079] which performs

interpolation during the animation process as set forth in the SVG standard, and *prima facie* the output would only be objects in the scene graph, and they would *prima facie* be based on the timeline as set forth in the rejection to claim 31 above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

53. As to claim 33,

The method of claim 32 wherein the composition engine is at a low-level with respect to the scene graph.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Steele in Fig. 15 shows that various programs, including the operating system, are on the flash memory, which in [0136] is specified to contain all the low-level programs of the operating system – graphics is low-level functionality. Since there is no specific graphics unit, all graphics operations and compositing are done by the operating system in the microprocessor, which *prima facie* means that in that embodiment, such graphics are done at a low-level, that is the rendering is done by the operating system at a low level. The scene graph is high-level in that it is embodied in RAM and is held as an abstraction – this is a function of the SVG standard that keeps

tree nodes and container nodes as abstract as possible, therefore the embodiment in Fig. 18 must do the same. In any case, low-level composition means that it would be done by hardware that is much faster than software. As such, it would be obvious to modify the device of Kim and Steele to use low-level rendering, and in any case Steele has the rendering means set forth in the rejection to claim 32 above, which is *prima facie* entirely hardware.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification such that rendering would be low-level. (Please see also rejection to claim 1 for additional motivation).

54. As to claim 34,

The method of claim 1 wherein causing data in the scene graph to be modified comprises invoking a function to place an object corresponding to audio and/or video data into the scene graph data structure.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Steele in Figs. 8 shows audio elements 820 and 830 in the animation execution and in Fig. 7 a scene graph data structure (a tree). Steele [0050, 0052] for example provides that such animation takes place, and the SVG standard in 6.18

clearly sets forth aural style sheets, that are audio data that each element can have animation associated with it, which clearly is placed into the scene graph of Fig. 7. Also, by definition, SVG animations would be video.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification and the audio capabilities of SVG. (Please see also rejection to claim 1 for additional motivation).

55. As to claim 35,

The method of claim 1 wherein causing data in the scene graph to be modified comprises changing a mutable value of an object in the scene graph data structure.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Steele teaches in [0014] that one embodiment of his invention changes the visual graph in accordance to changes of the sequence graph, where the visual graph is comparable to the "scene graph" of applicant and mutable values (e.g. position) of elements in the tree are shifted as per Steele [0052-0057]. Therefore, the limitation is met, and it would have been obvious to modify it so that it in fact change mutable values on the tree if applicant feels that this is not an adequate embodiment of this particular limitation.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification and the audio capabilities of SVG. (Please see also rejection to claim 1 for additional motivation).

56. As to claims 36 and 65,

[The means recited in claim 65 are those of the X3D parser of Kim and the SVG parser inherent in the Steele reference as set forth below, as applicant extensively references the SVG specification in his specification, and so the rejection on claim 36 is valid and binding on claim 65 without further comment.]

In a computing environment, a system comprising:

-A markup parser; (Kim Fig. 2, element 142 – parser, see [0025])(Steele inherently, because SVG graphics data requires DOM objects (see SVG standard), which inherently requires an XML parser as part of that system, as the XML data has to be processed and formatted, which is what a parser *by definition* does – as shown in Fig. 3, where the SVG comes into element 210 and then enters the SVG DOM 305 for translation from the SVG reader to the SVG compiler, which *prima facie* must have a parser (all compilers must parse syntax at a minimum – this is a fundamental of the computer art))

-An interface coupling the markup parser to a source of markup; and (interface is coupled to XML channel in Fig. 1 through communications network 150 and then into the X3D browser as shown in Fig. 3, which is shown to receive X3D data through communications module 130 and this is exactly stated in [0032])(Steele Fig. 3 shows the SVG input into element 210, which is the converter of Fig. 2, and a converter for data in this context – that is, data that is known to be a subset of XML **must** *prima facie* be parsed for conversion – Clearly, the converter receives function calls through the interface, e.g. the application 205 sends function calls to the converter for graphics data, which are then routed through the converter 210 – again, it is a fundamental of the computer art that applications sending data through other programs, particularly in this kind of a context.)

-A container for visual information of an object model, the markup parser converting markup received at the interface to method calls of objects in the object mode to modify data in the container for visual information. (SVG provides container objects for visual data, and SVG clearly teaches the use of container objects, as in section 1.6 it clearly teaches the use of 'container elements', which are defined as 'An element that can have graphic elements and other container elements as child elements'. Steele clearly teaches data modification in [0061] as set forth above. Clearly, the container object could be the head object of the tree structure shown in Steele Fig. 7.) (Further, X3D standard provides for container objects in the definitions, see "Container object", section 3).

Kim [0007-0008] clearly teaches the use of a scene graph and that X3D requires the construction of such scene graphs from primitives. Kim further teaches that the user can move through a scene [0020, 0026], which clearly establishes that a user is navigating and the scene is constantly being re-rendered, which *prima facie* requires data in the scene graph to be modified. Kim can also use MPEG-4, which clearly involves animation and modification of data in a scene graph. Steele uses SVG, and teaches the decomposition of graphic data into tree structures in Fig. 7. Further, Fig. 8 clearly illustrates what happens when the SVG animation language is transformed into two sets of output language data. Clearly, as Fig. 9 illustrates, animation is done with SVG on a routine basis and the translation is shown in Fig. 6. The tree of Fig. 7 is clearly a form of scene graph in the broad definition set forth above. Clearly, such a tree is a "scene graph". Thusly and *prima facie* obviously, the animation shown in Fig. 9 would cause data in a scene graph to be modified as the object was translated and the data structure containing locations and other information would changed.

Reference Steele clearly teaches how multimedia is converted and formatted for use for multiple devices. Reference Kim teaches the use of client 100 with X3D browser 110 over communication network 150 [0015, 0018, 0020]. Clearly, a portable device could fulfill this device (e.g. advanced PDA, cellular phone, or laptop.) Reference Steele teaches portable devices capable of performing advanced graphics functionality such as recited above in [0007-0008] and [0027], which clearly illustrates the invention functioning on a cellular phone in Fig. 27. Further, Steele teaches that his invention can be practiced on portable devices (including computers [0038]) – see Fig. 1

and the devices 105, which would clearly be the clients 100 of Kim and the communications networks would be comparable. Clearly, the references are directed to the same problem-solving area and further as set forth above are analogous art. As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize.

57. As to claim 37,

The system of claim 36 wherein the markup is converted to a method call to place a tree of visual objects into the scene graph data structure.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Further, Steele Fig. 6 shows an animation where path information is extracted into element 620, and listed in Fig. 7 [see Steele 0050 and 0079] as trees. SVG teaches that all implementations must implement a rendering model as set forth in 3.1 and so forth, and scene graphs are known to directed acyclic, i.e. trees. Clearly, Steele [0064] teaches that the visual graph is changed according to the addition or alteration of elements in the sequence graph, which clearly would happen with the insertion of new visible objects into the tree shown in Fig. 7, i.e. during animation when new elements were being shown. It would have been obvious to modify the system of Steele to so perform the recited task.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality and perform the recited conversion. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

58. As to claim 38,

The system of claim 36 wherein the markup is converted to a method call to arrange a tree of visual objects in the scene graph data structure.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Further, Steele Fig. 6 shows an animation where information is extracted and separated into specific elements, and shown / listed in Figs. 7 and 8 [see Steele 0050 and 0079] as trees. SVG teaches that all implementations must implement a rendering model as set forth in 3.1 and so forth, and scene graphs are known to directed acyclic, i.e. trees. Clearly, Steele [0064] teaches that the visual graph is changed according to the addition or alteration of elements in the sequence graph, which clearly would happen with the insertion of new visible objects into the tree shown in Fig. 7, i.e. during animation when new elements were being shown.

Specifically, the arrangement of a tree in the scene graph data structure again would be dependent on the insertion of items into the visual tree, and obviously given that Steele and SVG teach container objects as taught in the rejection to claim 36

above, the insertion of a container object would *prima facie* cause the insertion of a tree, and arranging a tree in the scene graph would qualify as "altering" elements within the system of Steele and falls within [0064].

It would have been obvious to modify the system of Kim in light of Steele to so perform the recited task.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality and perform the recited conversion. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

59. As to claim 39,

The system of claim 37 wherein the markup is converted to a method call to place a visual collection object into the scene graph data structure.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Further, Steele Fig. 6 shows an animation where information is extracted and separated into specific elements, and shown / listed in Figs. 7 and 8 [see Steele 0050 and 0079] as trees. SVG teaches that all implementations must implement a rendering model as set forth in 3.1 and so forth, and scene graphs are known to directed acyclic, i.e. trees. Clearly, Steele [0064] teaches that the visual graph is changed according to the addition or alteration of elements in the sequence graph,

which clearly would happen with the insertion of new visible objects into the tree shown in Fig. 7, i.e. during animation when new elements were being shown.

--A collection object would clearly encompass a container object containing a tree of visual objects; examiner is interpreting the definition in this way.

Specifically, the arrangement of a tree in the scene graph data structure again would be dependent on the insertion of items into the visual tree, and obviously given that Steele and SVG teach container objects as taught in the rejection to claim 36 above, the insertion of a container object would *prima facie* cause the insertion of a tree, and arranging a tree in the scene graph would qualify as "altering" elements within the system of Steele and falls within [0064].

It would have been obvious to modify the system of Kim in light of Steele to so perform the recited task.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality and perform the recited conversion. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

60. As to claim 40,

The system of claim 36 wherein the markup is converted to a method call to place a visual object into the scene graph data structure.

Reference Kim does not expressly teach this limitation, but clearly teaches that users navigate through a virtual environment, which *prima facie* requires that transforms take place to visual objects and new visual objects be inserted (see rejections to claim 2 and 3). Reference Steele teaches this limitation, as for example he teaches the insertion of unique identifiers into media streams [0106], and further [0088] that any visual element or object can modified. Such modifications and insertions *prima facie* must associate code with a suitable / desired insertion as that is the only way either a hierarchy of nodes (e.g. Fig. 7) or single nodes could be logically inserted.

For the second case, if the definition of context is the data associated with a specific element – e.g. the details of the element, its location, color, et cetera, these attributes are inherent in SVG elements as set forth in the rejections above, e.g. sections 11.1, 9.1-9.7, et cetera. Further, Steele teaches the same in Figure 7, where each element has certain properties that would be a drawing context, in the sense that each visual element has those properties associated with it [Steele 0052-0056 and 0059-0061].

As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize. (Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task.)

61. As to claim 41,

The system of claim 40 wherein the markup is converted to a method call to associate a brush with the visual object.

This claim is a substantial duplicate of claim 6. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 6 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. The rejection to claim 6 is herein incorporated by reference with the motivation and combination. The SVG standard is referenced there so there is no difference in the reference hierarchy and it would have been trivially obvious to combine as set forth because the SVG specification is extensively cited by and key to the Steele reference.

62. As to claim 42,

The system of claim 40 wherein the markup is converted to a method call to associate a geometry with the visual object.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in the rejection to claim 36 and reference SVG clearly supports this position. The SVG specification sets forth classes of shapes in section 9.1, where all six of the recited shapes (rectangle, polygon, path, line, polyline, and ellipse) are set forth. Further, the SVG view in Steele decomposes SVG instructions into scene graphs containing basic SVG shapes as above [Steele 0052], where 'Visual Elements' include Shape classes. SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup. Secondarily, the

association of a geometry with a visual object would be intrinsic to the existence of such an object. However, SVG in section 11.4 sets forth different shapes of strokes and the same in 11.6 for markers. Therefore, it would have been obvious to one of ordinary skill in the art to modify the Kim reference in light of Steele to be able to change the type of geometry associated with a visual element in accordance with the teachings of SVG 11.4 and 11.6, which meets the above-recited limitations.

As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 36 for additional motivation).

63. As to claim 43,

The system of claim 42 wherein the geometry comprises at least one of a set containing an ellipse geometry, a rectangle geometry, a line geometry and a path geometry.

This claim is a substantial duplicate of claim 13. Further, the term 'geometry' used here is analogous to the 'shape' reference in claim 13. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 13 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. As such, the rejection for claim 13 is hereby incorporated by reference with motivation and combination and is valid without further comment. The six components of the SVG standard shapes include an ellipse, a rectangle, a line, and a path – see SVG section 9.1.

64. As to claim 44,

The system of claim 40 wherein the markup is converted to a method call to associate a transform with the visual object.

This claim is a substantial duplicate of claim 3. Further, the term 'geometry' used here is analogous to the 'shape' reference in claim 3. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 3 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. As such, the rejection for claim 3 is hereby incorporated by reference with motivation and combination and is valid without further comment. The six components of the SVG standard shapes include an ellipse, a rectangle, a line, and a path – see SVG section 9.1.

65. As to claim 45,

The system of claim 44 wherein the transform comprises a rotate transform for changing a perceived angle of the visual object.

This claim is a substantial duplicate of claim 3. Further, the term 'geometry' used here is analogous to the 'shape' reference in claim 3. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 3 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art.

Specifically, the rejection to claim 3 cites Steele [0053], which clearly states that rotations are applied to visual objects or groups of visual objects, and states that as well.

As such, the rejection for claim 3 is hereby incorporated by reference with motivation and combination and is valid without further comment. The six components of the SVG standard shapes include an ellipse, a rectangle, a line, and a path – see SVG section 9.1.

66. As to claim 46,

The system of claim 44 wherein the transform comprises a scale transform for changing a perceived size of the visual object.

Reference Kim does not expressly teach this limitation, but clearly teaches that users navigate through a virtual environment, which *prima facie* requires that transforms take place to visual objects. Reference Steele teaches this implicitly limitation, as he discloses rotations in [0053] and further states that rotations and other transformations can be applied to an entire tree of objects, e.g. Fig. 7, and further [0088] that any visual element or object can modified. Such modifications *prima facie* must associate code with a suitable / desired transform (e.g. scaling, rotation, et cetera [0053]), as that is the only way either a hierarchy of nodes (e.g. Fig. 7) or single nodes could be scaled.

Specifically, however, the SVG specification provides for scaling transformations in Example Rotate-Scale in section 7.4 under coordinate system transformations

As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG specification, and because they provide for coordinate transforms as set forth above, and the Kim reference inherently performs transforms as a user can navigate through a virtual 3D

environment, which inherently requires graphics transforms and such. (Please see claim 36 for additional motivation / combination justification, which is herein incorporated by reference). (Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task.)

67. As to claim 47,

The system of claim 44 wherein the transform comprises a translate transform for changing a perceived position of the visual object.

This claim is a substantial duplicate of claim 3. Further, the term 'geometry' used here is analogous to the 'shape' reference in claim 3. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 3 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art.

Specifically, the rejection to claim 3 cites Steele [0053], which clearly states that translations are applied to visual objects or groups of visual objects, and states that as well.

As such, the rejection for claim 3 is hereby incorporated by reference with motivation and combination and is valid without further comment. The six components of the SVG standard shapes include an ellipse, a rectangle, a line, and a path – see SVG section 9.1.

68. As to claim 48,

The system of claim 44 wherein the transform comprises a skew transform for changing a perceived skew of the visual object.

Reference Kim does not expressly teach this limitation, but clearly teaches that users navigate through a virtual environment, which *prima facie* requires that transforms take place to visual objects. References Steele and SVG teach this implicitly, that is Steele teaches it implicitly, this limitation that is, as he discloses rotations in [0053] and further states that rotations and other transformations can be applied to an entire tree of objects, e.g. Fig. 7, and further [0088] that any visual element or object can modified. Such modifications *prima facie* must associate code with a suitable / desired transform (e.g. scaling, rotation, et cetera [0053]), as that is the only way either a hierarchy of nodes (e.g. Fig. 7) or single nodes could be scaled.

Specifically, however, the SVG specification provides for skew transformations in Example Skew in section 7.4 under coordinate system transformations.

As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG specification, and because they provide for coordinate transforms as set forth above, and the Kim reference inherently performs transforms as a user can navigate through a virtual 3D environment, which inherently requires graphics transforms and such. (Please see claim 36 for additional motivation / combination justification, which is herein incorporated by reference). (Further, note that since this is performed by software,

prima facie 'code' that is software elements, would be invoked to perform any recited task.)

69. As to claim 49,

The system of claim 44 wherein the markup provides animation information associated with the transform, and wherein the animation information causes transformation data associated with the transform to change over time thereby animating the transformation of the visual object over time.

Reference Kim does not expressly teach this limitation, while references Steele and SVG do teach it. Steele clearly establishes in [0051-0054] and Figs. 6 and 9 that animation takes place through the SVG standard. Section 19.2 of SVG sets forth how this takes place, and at the bottom three paragraphs of section 19.2.2 it clearly states that animati0on has a document start and document end, and further in the second to last paragraph that the SVG system indicates the timeline position of document fragments being animated. Further, according to SVG 19.2.2 the animation is by document fragment and object path, which clearly are passed to the system is specified in, for example, the second page of Fig. 9 in the SVG XML program. Clearly, the system of Steele performs compositing and rendering [0007, 0011-0012], as does Kim. Finally, reference SVG teaches that it supports compositing (section 14.2.1), and the X3D specification supports compositing (6.2.3 for example). The composition engine would be, for example, elements 143, 144, and 147 of Fig. 2 of Kim.

Steele in Figs. 6 and 9 shows animated elements [0041] and in Fig. 8, clearly during an animation the actions are shown, where the system in steps 835, 840, and

845 performs interpolation for nodes shown in the tree in Fig. 7. Clearly interpolation takes place during animation [0072, 0077-0079] which performs interpolation during the animation process as set forth in the SVG standard, and *prima facie* the output would only be objects in the scene graph, and they would *prima facie* be based on the timeline as set forth above.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claim 1 for additional motivation).

70. As to claim 50,

The system of claim 40 wherein the markup is converted to a method call to associate a color with the visual object.

This claim is a substantial duplicate of claim 7. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 3 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. A color as recited here is applied via the paint method (SVG 11.1 – 11.4) and is the exact same as the "solid color" associated with a "brush" as in claim 7 – the "brush" is merely a semantic description as set forth above. As such, the rejection for claim 7 is hereby incorporated by reference with motivation and

combination and is valid without further comment. Clearly, SVG teaches the use of a solid color with a visual object – see SVG 11.1.

71. As to claim 51,

The system of claim 40 wherein the markup is converted to a method call to associate gradient data with the visual object.

This claim is a substantial duplicate of claim 7. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 3 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. A gradient as recited here is applied via the paint method (SVG 11.1 – 11.4) and is the exact same as the "solid color" associated with a "brush" as in claim 7 – the "brush" is merely a semantic description as set forth above, and a linear gradient is certainly a gradient. As such, the rejection for claim 7 is hereby incorporated by reference with motivation and combination and is valid without further comment. Clearly, SVG teaches the use of a gradient with a visual object – see SVG 11.1.

72. As to claim 52,

The system of claim 40 wherein the markup is converted to a method call to associate a tile brush with the visual object.

Reference Kim does not expressly teach this limitation. Reference Steele does teach this limitation by the use of SVG graphics. Turning to the SVG (, section 11 titled 'Painting: Filling, Stroking, and Marker Symbols', specifically section 11.1, 'With SVG, you can paint (e.g. stroke or fill) with: ..." and then proceeds to list several. The term 'brush data' is clearly analogous to the 'paint' operation in SVG with comparable data.

Given that SVG allows (from section 11.1), among other things a pattern (vector or image) that can be tiled, clearly each visible element clearly has such data associated with it (see section 11.2 in its entirety, 11.7 for specific properties, section 11.8 for how painting properties can be inherited, which *prima facie* justifies the position that element have intrinsic painting properties, i.e. brush data as set forth above. Further, note that since this is performed by software, *prima facie* 'code' that is software elements, would be invoked to perform any recited task, and SVG data element *prima facie* and inherently possess paint data as set forth by the SVG specification above. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification, and because they serve complementary and supplementary purposes in how they handle graphics and animation, particularly with respect to the standards they utilize, and the SVG standard inherently handles these paint limitations.

73. As to claim 53,

The system of claim 40 wherein the markup is converted to a method call to associate an image with the visual object.

This claim is a substantial duplicate of claim 10. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 3 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. An image as recited here is applied via the paint method (SVG

11.1 – 11.4) and is the exact same as the image associated with a "brush" as in claim 7 – the "brush" is merely a semantic description as set forth above. As such, the rejection for claim 10 is hereby incorporated by reference with motivation and combination and is valid without further comment. Clearly, SVG teaches the use of an image with a visual object – see SVG 11.1.

74. As to claim 54,

The system of claim 40 wherein the markup is converted to a method call to associate a drawing comprising drawing primitives with the visual object.

References Kim and Steele do not expressly teach these limitations; reference Steele teaches them intrinsically as set forth above in claim 1 and reference SVG clearly supports this position. The SVG specification sets forth classes of shapes in section 9.1, where six shapes are set forth, and they are fundamental drawing primitives. Also, in SVG 7.11, the table lists a "primitiveUnits" that stand for numeric primitives within elements of the basic classes or elements being referenced. . Further, the SVG view in Steele decomposes SVG instructions into scene graphs containing basic SVG shapes as above [Steele 0052], where 'Visual Elements' include Shape classes. SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification. (Please see also rejection to claims 1 and 2 for additional motivation and notes).

75. As to claim 56,

The system of claim 40 wherein the markup is converted to a method call to associate audio and/or video media with the visual object.

References Kim does not expressly teach this limitation, while reference Steele does teach it. Steele in Figs. 8 shows audio elements 820 and 830 in the animation execution and in Fig. 7 a scene graph data structure (a tree). Steele [0050, 0052] for example provides that such animation takes place, and the SVG standard in 6.18 clearly sets forth aural style sheets, that are audio data that each element can have animation associated with it, which clearly is placed into the scene graph of Fig. 7. Also, by definition, SVG animations would be video.

SVG is a markup language, therefore any SVG rendering utility would *prima facie* receive markup, and any SVG rendering program would *prima facie* invoke such functionality. As such, reference Steele intrinsically teaches this limitation and it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG standard specification and the audio capabilities of SVG. (Please see also rejection to claim 1 for additional motivation).

76. As to claim 57,

The system of claim 40 wherein the markup is converted to a method call to associate an image effect with the visual object.

This claim is a substantial duplicate of claim 11. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 3

merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. An image effect as recited here is applied as set forth in SVG section 14.4, with the alpha blending of course being an image effect. As such, the rejection for claim 11 is hereby incorporated by reference with motivation and combination and is valid without further comment, as the references are the same. Clearly, SVG teaches the use of an image with a visual object – see SVG 14.4.

77. As to claim 58,

The system of claim 40 wherein the markup is converted to a method call to associate a pen with the visual object, to describe how a shape is outlined.

This claim is a substantial duplicate of claim 12. The only difference is that the markup is converted to a method call to perform the recited action, whereas claim 12 merely 'invokes code' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. . As such, the rejection for claim 12 is hereby incorporated by reference with motivation and combination and is valid without further comment, as the references are the same. Clearly, SVG teaches the use of a pen to set outlines – see SVG 11.4 and 11.5.

78. As to claim 59,

The system of claim 40 wherein the markup is converted to a method call to obtain a drawing context associated with the visual object.

This claim is a substantial duplicate of claim 26. The only difference is that the markup is converted to a method to call to perform the recited action, whereas claim 12 merely makes a 'function call' to perform the recited task, and as such would a trivial

variation of ordinary skill in the art. As such, the rejection for claim 26 is hereby incorporated by reference with motivation and combination and is valid without further comment, as the references are the same.

79. As to claim 60,

The system of claim 40 wherein the markup is converted to a method call to associate hit testing data with the visual object.

This claim is a substantial duplicate of claim 17. The only difference is that the markup is converted to a method to call to perform the recited action, whereas claim 17 merely 'causes data to be modified' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. As such, the rejection for claim 17 is hereby incorporated by reference with motivation and combination and is valid without further comment, as the references are the same.

80. As to claim 61,

The system of claim 40 wherein the markup is converted to a method call to associate a rectangle with the visual object.

This claim is a substantial duplicate of claim 14. The only difference is that the markup is converted to a method to call to perform the recited action, whereas claim 14 merely 'causes data to be modified' to perform the recited task, and as such would a trivial variation of ordinary skill in the art. As such, the rejection for claim 14 is hereby incorporated by reference with motivation and combination and is valid without further comment, as the references are the same. Also, clearly whether or not the visual

object is in the scene graph does not matter in the context of this specific claim as it is in claim 14.

81. As to claim 61,

The system of claim 61 wherein the markup includes data describing how a source rectangle should be stretched to fit a destination rectangle.

Reference Kim does not expressly teach this limitation, but clearly teaches that users navigate through a virtual environment, which *prima facie* requires that transforms take place to visual objects. Reference Steele teaches this implicitly limitation, as he discloses rotations in [0053] and further states that rotations and other transformations can be applied to an entire tree of objects, e.g. Fig. 7, and further [0088] that any visual element or object can modified. Such modifications *prima facie* must associate code with a suitable / desired transform (e.g. scaling, rotation, et cetera [0053]), as that is the only way either a hierarchy of nodes (e.g. Fig. 7) or single nodes could be scaled.

Specifically, however, the SVG specification provides for scaling transformations in Example Rotate-Scale in section 7.4 under coordinate system transformations. Further, SVG clearly teaches the use of rectangles as basic shapes and provides for methods for associating one shape with a visual object as set forth in SVG 9.1. Obviously from the parent claim, a rectangle could be scaled using the methods set forth in the SVG standard listed immediately above and it would have been obvious to one ordinary skill in the art to so modify the apparatus of Kim in light of Steele and SVG as set forth above.

As such, It would have been obvious to one having ordinary skill in the art at the time the invention was made to combine the X3D and graphics system of Steele with the SVG and graphics system of Kim as set forth above and the SVG specification, and because they provide for coordinate transforms as set forth above, and the Kim reference inherently performs transforms as a user can navigate through a virtual 3D environment, which inherently requires graphics transforms and such. (Please see claim 61 for additional motivation / combination justification, which is herein incorporated by reference).

Conclusion

82. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure:

-US Patent 6,714,242 to Itou;

-US PGPub 20030005045 to Tanimoto;

-US PGPub 20030110297 to Tabatabai et al;

-US PGPub 20030126557 to Yardumian et al.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Eric V Woods whose telephone number is 703-305-0263. The examiner can normally be reached on M-F 7:30-5:00 alternate Fridays off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Michael Razavi can be reached on 703-305-4713. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Eric Woods January 19, 2005

JEFFREY BRIER
PRIMARY EXAMINER